

Heuristic-based Optimal Resource Provisioning in Application-centric Cloud

Sunirmal Khatua, *Member, IEEE*, Preetam K. Sur, Rajib K. Das and Nandini Mukherjee, *Senior Member, IEEE*

Abstract—Cloud Service Providers (CSPs) adapt different pricing models for their offered services. Some of the models are suitable for short term requirement while others may be suitable for the Cloud Service User's (CSU) long term requirement. In this paper, we look at the problem of finding the amount of resources to be reserved to satisfy the CSU's long term demands with the aim of minimizing the total cost. Finding the optimal resource requirement to satisfy the CSU's demand for resources needs sufficient research effort. Various algorithms were discussed in the last couple of years for finding the optimal resource requirement but most of them are based on IPP which is NP in nature. In this paper, we derive some heuristic-based polynomial time algorithms to find some near optimal solution to the problem. We show that the cost for CSU using our approach is comparable to the solution obtained using optimal Integer Programming Problem(IPP).

Index Terms—Cloud Computing, Amazon EC2, Reserved Instance, Heuristics, Cost Optimization

1 INTRODUCTION

Recent advances in distributed computing has brought a paradigm shift in the computing world. The practices of dedicated access to computers owned by individuals or organizations have been replaced by on-demand accesses to resources shared among many individuals and different organizations. Cloud Computing has largely contributed to this transition by enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [4]

Present practice in the Cloud Computing arena is that the Cloud Service Users (CSUs) request for resources and Cloud Service Providers (CSPs) allot the virtualised resources to the CSUs according to their requirements. However, while requesting the resources, CSUs face a major challenge due to various pricing schemes offered by the CSPs. Resources are available on reservation basis, as well as on demand basis. Reservation is to be done for a fixed contract period with a fixed price. However, reservation for a longer period or for larger amount of resources than that required to meet the demand for resources of an application, may lead to over-provisioning and higher cost for the CSUs. On the other hand, on-demand prices are generally higher than the reserved one and therefore if the resources are allotted only on-demand basis, cost to be paid by the CSUs will again be high.

Therefore, some optimization strategies are required to reduce the resource usage charges from the users' point of view.

This paper focuses on the optimizing strategies aimed at reducing the total cost of cloud deployment. Remaining part of the paper is organised as follows. Various pricing schemes offered by different CSPs are discussed in the next section. A brief overview of the related work is given in section 3. An IPP based optimal resource provisioning strategy for the application-centric cloud framework is discussed in Section 4. Section 5 and 6 present the proposed heuristics for polynomial time resource reservation strategies. The performance of the heuristic based approach compared to the IPP is presented in Section 7 and section 8 concludes the paper.

2 PRICING SCHEMES

Cloud computing service providers offer computing resources as utility and software as a service over a network. Cloud users pay for these resources or services on the basis of their usage. Optimising the cost of resources and services from the *Cloud Service User's* point of view is a tricky issue as the CSPs often provide non-fixed pricing models for their services. A brief description of the general trend in the pricing models offered by the CSPs is given below.

- 1) Fixed Costs - Resources are charged by its type and duration (e.g. month, year etc). Here exactly one price is assigned for a fixed time duration (say, for a month). The cost is computed as the fixed charge multiplied by the number of resource instances or service units requested by the consumer. The users pay the fixed cost even though the resources may not be used for the entire time duration.
- 2) Variable Costs - Under variable cost model, Cloud computing delivers computing power and services to consumers on a variable cost pay-as-you-go basis determined by the number of users and their volume of

- S. Khatua and R.K.Das are with the Department of Computer Science and Engineering, University of Calcutta, India.
E-mail: skhatuacomp@caluniv.ac.in and rajib.das.k@gmail.com
- P. K. Sur is with the Jadavpur University, Kolkata, India.
E-mail: mailtopreetam@gmail.com
- Nandini Mukherjee is with the Department of Computer Science and Engineering, Jadavpur University, India
E-mail: nmukherjee@cse.jdvu.ac.in

transactions. Resources are charged by their types and usage (e.g. per hour) and with no long-term commitments or upfront payments. Here the CSUs need to pay on different items and deal with several different structures. Resources are allotted on-demand basis and the users do not need to pay if the resources are not used.

- 3) Hybrid Costs - Hybrid Costs are a mixed form of the former two cost structures, where one can find a variable part and a fixed part.
- 4) Flexible Costs - Resources are charged by its type and time of usage. At a specific instance of time, the cost of the resource is set depending on the demand of the resources. Like variable costs, users do not need to pay if they do not use the resources.

In this paper, we have used the pricing schemes offered by Amazon as a case study. Amazon EC2, one of the major IaaS (Infrastructure As A Service) provider, offer pricing schemes in the form of On-Demand, Reserved and Spot instances. On-Demand instances support variable costs having a fixed hourly charge of usage. Reserved instances support the hybrid cost model with a fixed one time reservation cost, which depends on the contract duration (e.g. 1 year or 3 years) and a hourly usage cost which is a discounted price over the cost of the On-Demand instances. Flexible costs are supported by the Spot Instances which are similar to On-Demand but having flexible hourly charge of usage. So cost optimization can be considered on Spot instances, Reserved instances, as well as on both. In this paper, we have not considered Spot instances for cost minimization.

3 RELATED WORK

Various aspects of multiple pricing schemes have been studied in [11]-[15]. Most of these research works consider reserved versus on-demand pricing scheme. A model for determining the price of a reserved instance as well as an on-demand instance is proposed in [11]. The research work presented in [12] considers the revenue maximization problem from a CSP's point of view, separating the CSUs into 'premium' (with reserved services) and 'basic' (with on-demand services) users. Both papers demonstrate the effectiveness of having multiple pricing schemes, such as reservation and on-demand which are followed by most of the CSPs now-a-days.

Cost optimization from a CSU's point of view, by considering different pricing schemes, have been studied in [13], [14] and [15]. A stochastic integer programming model has been adopted in [13] to optimize the cost of SLA-aware resource scheduling in cloud. H. Menglan et al., in their paper [14], consider on-demand and reserved instances to optimize cost for deadline constrained jobs and to optimize total processing time for budget constrained jobs. An optimal cloud resource provisioning (OCRP) algorithm is proposed by S. Chaisiri et al. in their paper [15]. They have also considered a stochastic integer programming model with demand and price uncertainty in the OCRP algorithm. They have extended their work by considering spot pricing scheme in [15]. These solutions need the predicted demands for resources. Demand

forecasting models for twelve months have been developed in [16] based on historical data.

Most of these previous works for optimizing cost use some form of integer programming model which is NP in nature. They have solved the problem using some techniques like benders decomposition and sample-average approximation as have been used in [15]. To the best of our knowledge, no one has used some good heuristics to solve the cost optimization problem in polynomial time. Motivated by these works, we have developed some heuristics to solve the cost optimization problem in linear time. We have also mathematically proved that our heuristic provides optimal solution in certain scenarios. However, for simplicity, we do not consider uncertainty of demands and prices in this work.

4 OPTIMAL RESOURCE PROVISIONING IN APPLICATION-CENTRIC CLOUD

In our previous work [3], we have proposed a resource provisioning framework for application-centric cloud as shown in Figure 1. Two key components of the framework are the

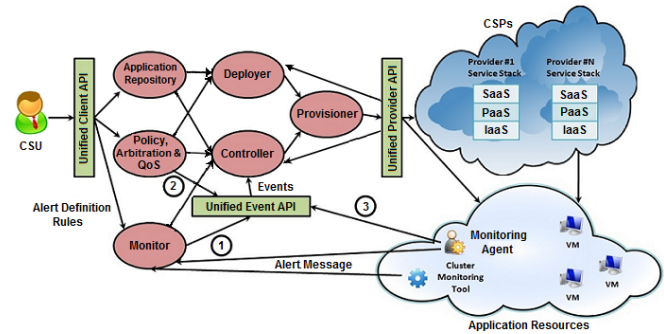


Fig. 1. Resource Provisioning Framework for Application-centric Cloud

Deployer module and the Controller module. The Deployer module statically analyzes an application to determine the optimal resource requirement for the application. The Controller module dynamically fine-tunes the provisioned resources to alleviate the overprovisioning and underprovisioning situations.

In this paper, we extend the functionality of the Deployer module to statically determine the amount of resources to be reserved in advance in order to minimize the total cost of running an application. While doing this, we make the following assumptions -

- An application is run through different stages denoted by $t, 1 \leq t \leq T$. The granularity of a stage is determined by the number of hours per stage (h).
- At every stage of execution of the application, its demand for resources is known (or estimated using the mechanisms described in [16]) and is denoted by the vector D . Since the reservation made at any particular time has a long term effect, in order to find an optimal reservation schedule, the future value of demand for resources is to be known in advance. It is therefore assumed that the predicted values of the demand vector are available at each stage $t, 1 \leq t \leq T$.

- Resources can be reserved under different contracts (e.g. for a 1 year contract or for a 3 year contract etc.)
- Reserved resources have a one-time fixed charge for the duration of the contract and a variable usage charge at discounted rate to be paid on hourly basis.
- Resource instances can be reserved and launched at the beginning of a stage and can be terminated at the end of a stage. If the demand at a given stage t is more than the total amount of resources reserved, the balance requirement is satisfied with on-demand instances.

Thus, the optimal resource reservation problem can be formally stated as

Given a vector of demands D for resources at different stages t , $1 \leq t \leq T$, find the number of resources to be reserved under different contracts at each stage t , so that the total cost is minimized satisfying the demands at every stage.

The above problem can be formulated as an Integer Programming Problem(IPP), which is discussed below.

4.1 IPP-based Optimal Resource Reservation

This section discusses an IPP formulation to find the optimal resource reservation schedule to minimize cost. The various notations used in the formulation are given in Table 1. There are K different types of reservation contracts offered by a CSP. Each type of contract (k) is associated with a one time reservation cost (R_k), usage cost (r_k) per hour and its duration (t_k) in number of stages. At every stage (t), we need to decide how much instances to be reserved ($x_t^{R_k}$). We also need to determine the number instances to be launched based on reservation from contract k ($x_t^{r_k}$) as well as on-demand (x_t^o). The cost of on-demand instances is more than the usage cost of reserved instances ($o > r_k$). At the same time, increasing reservation incurs high reservation cost. So, the objective is to find the optimal reservation balancing these two factors.

TABLE 1
Definitions and Notations

Symbol	Definition
K	Number of reservation contracts offered by a CSP.
T	Total number of stages for which the demands are available.
h	Duration of each stage in hours
D_t	The demand for instance-hours at stage t , $1 \leq t \leq T$.
R_k	One time reservation charge per instance under contract k .
t_k	Duration of contract k in stages, $1 \leq k \leq K$.
r_k	Usage charge per hour for a reserved instance under contract k .
o	Usage charge per hour for an on-demand instance.
$x_t^{R_k}$	Number of instances reserved under contract k at stage t .
$x_t^{r_k}$	Number of reserved instances launched under contract k at stage t .
x_t^o	Number of ondemand instances launched at stage t .

4.2 IPP Formulation

In this section, an integer programming problem is formulated with an objective of optimal resource provisioning. Based on the previous definitions, the cost at any given stage t is denoted by C_t , where

$$C_t = \sum_{k=1}^K (x_t^{R_k} \cdot R_k + x_t^{r_k} \cdot r_k \cdot h) + x_t^o \cdot o \cdot h :$$

The first term in the above expression stands for the cost of reservation under contract k , the second term stands for the cost of using reserved instances, and the last term stands for the cost of using on-demand instances.

Our objective is to minimize the total cost over all stages with the following constraints -

- 1) Number of reserved instances and number of instances launched from reserved instances as well as on-demand instances at any stage are non-negative integers.
- 2) Instances launched at any stage t under contract k cannot exceed the total reserved instances under contract k . It is to be noted that reservation under contract k made at stage t is effective up to stage $t + t_k - 1$. In other words, only the instances which have been reserved at stage $t - t_k + 1$ or after that are available at stage t .
- 3) At any stage t , total number of instances launched under all the contracts plus on demand instances must satisfy the demand of the application.

Thus, the integer programming problem is as follows:

Minimize $\sum_{t=1}^T C_t$ subject to the following conditions

- i) $x_t^{R_k}, x_t^{r_k}, x_t^o \geq 0$ and integer
- ii) $\sum_{i=t-t_k+1, i \geq 1}^t x_i^{R_k} \geq x_t^{r_k} \quad \forall k = 1, 2, \dots, K \text{ \& } \forall t = 1, 2, \dots, T$
- iii) $x_t^o + \sum_{k=1}^K x_t^{r_k} \geq D_t \quad \forall t = 1, 2, \dots, T$

5 HEURISTIC-BASED RESOURCE RESERVATION

An IPP is NP-hard and therefore, no polynomial time algorithm does exist to solve the optimal resource reservation problem as discussed in Section 4. In this section, we derive some heuristics for solving the optimal resource reservation problem in polynomial time. We show that when there is a single type contract k with contract duration of t_k stages, and the demand vector is available for stages $t = 1, \dots, t_k$, it is possible to determine the optimal value for reservation under contract k . The discussion in this section is based on the assumption that *duration of each stage is 1 hour*.

The usage cost for a demand d in a single stage (1 hour) with x reserved instances under contract k is:

$$\begin{aligned} \zeta(x, d) &= \begin{cases} d \cdot r_k & \text{if } x \geq d \\ x \cdot r_k + (d - x) \cdot o & \text{otherwise} \end{cases} \\ &= r_k \cdot \min(x, d) + o \cdot \max(d - x, 0) \end{aligned} \quad (1)$$

Let us assume that the vector of demands D for the duration of t_k stages is available. Let x be the amount of resources reserved under contract k and let E_x denotes the total cost corresponding to demand vector D , that includes both, cost of reservation and cost of using the resources. Thus, combining equation 1 with the cost of reservation, E_x can be written as

$$E_x = x.R_k + \sum_{i, D_i \leq x} D_i.r_k + \sum_{i, D_i > x} (x.r_k + (D_i - x).o) \quad (2)$$

Let D^s be the demand vector obtained by sorting D in ascending order. Considering $x = D_j^s$, Equ. 2 can be rewritten as

$$E_{D_j^s} = D_j^s.R_k + \sum_{i=1}^j D_i^s.r_k + \sum_{i=j+1}^{t_k} (D_j^s.r_k + (D_i^s - D_j^s).o) \quad (3)$$

Now by replacing the on-demand cost o with $r_k + \alpha_k$, where $\alpha_k = o - r_k$ is the hourly discount for using reserved instances under contract k over on-demand instances, we have

$$\begin{aligned} E_{D_j^s} &= D_j^s.R_k + \sum_{i=1}^j D_i^s.r_k + \\ &\quad \sum_{i=j+1}^{t_k} [D_j^s.r_k + (D_i^s - D_j^s).(r_k + \alpha_k)] \\ &= D_j^s.R_k + \sum_{i=1}^j D_i^s.r_k + \sum_{i=j+1}^{t_k} (D_i^s - D_j^s).\alpha_k \end{aligned} \quad (4)$$

If the amount of reserved resources is increased to the next higher value in the sorted demand vector, (i.e. D_{j+1}^s as shown in Fig. 2), then the total cost is given by equation 5.

$$E_{D_{j+1}^s} = D_{j+1}^s.R_k + \sum_{i=1}^{t_k} D_i^s.r_k + \sum_{i=j+2}^{t_k} (D_i^s - D_{j+1}^s).\alpha_k \quad (5)$$

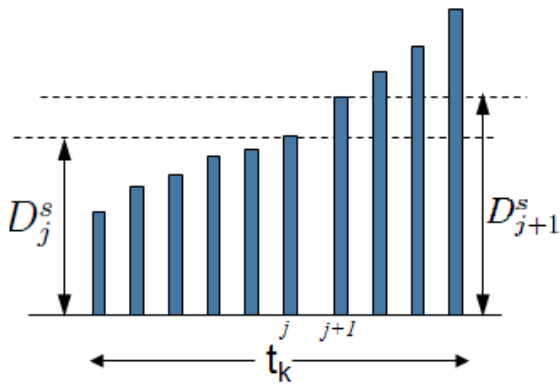


Fig. 2. Increasing reservation from D_j^s to D_{j+1}^s .

Thus, the change in total expenditure due to change in reservation from D_j^s to D_{j+1}^s is

$$\begin{aligned} \Delta E_{D_j^s, D_{j+1}^s} &= (D_{j+1}^s - D_j^s).R_k - \sum_{i=j+2}^{t_k} ((D_{j+1}^s - D_j^s).\alpha_k - \\ &\quad (D_{j+1}^s - D_j^s).\alpha_k) \\ &= (D_{j+1}^s - D_j^s).R_k - (D_{j+1}^s - D_j^s).\alpha_k(t_k - j) \\ &= (D_{j+1}^s - D_j^s)(R_k - t_k.\alpha_k + j.\alpha_k) \end{aligned} \quad (6)$$

Now we state the following two lemmas based on the characteristics of reserved instances and equation 6.

Lemma 1: For any contract type k , $R_k < t_k.\alpha_k$.

Proof: Consider a situation where demand is constant, i.e., $D_j = x$ for $1 \leq j \leq t_k$. If we reserve the amount x , the cost is $R_k.x + r_k.t_k.x$ and there is no on demand charge. If we make no reservation then cost is $o.t_k.x$. The second option is costlier, because if provisioning using only on-demand instances is cheaper, then no one would go for reservation. Thus, $o.t_k.x > R_k.x + r_k.t_k.x$, which implies $t_k.\alpha_k > R_k$, where $o = r_k + \alpha_k$ \square

Lemma 2: The number of instances to be reserved, for which the total cost is minimized, is always a member of the demand vector.

Proof: We prove the above lemma by contradiction. If possible, let us consider the existence of p , $D_j^s < p < D_{j+1}^s$, such that $E_{D_j^s} > E_p$ and $E_{D_{j+1}^s} > E_p$.

Now, we have,

$$\begin{aligned} E_p &= p.R_k + \sum_{i=1}^j D_i^s.r_k + \\ &\quad \sum_{i=j+1}^{t_k} [p.r_k + (D_i^s - p)(r_k + \alpha_k)] \text{ and} \\ \Delta E_{D_j^s, p} &= (p - D_j^s)(R_k - t_k.\alpha_k + j.\alpha_k) \\ &= (p - D_j^s).f(j, k) \end{aligned}$$

where, the expression $R_k - t_k.\alpha_k + j.\alpha_k$ is denoted by $f(j, k)$. Similarly, it can be shown that

$$\Delta E_{p, D_{j+1}^s} = (D_{j+1}^s - p).f(j, k)$$

As $D_j^s < p < D_{j+1}^s$, $\Delta E_{D_j^s, p}$ and $\Delta E_{p, D_{j+1}^s}$ will have the same sign which will depend on the sign of $f(j, k)$. But, this contradicts our initial assumption that $E_{D_j^s} > E_p$ and $E_{D_{j+1}^s} > E_p$. Hence, the proof. \square

Based on lemma 1 and lemma 2, the following theorem is stated.

Theorem 1: For a single type contract k , with demand vector D available for $t = 1, 2, \dots, t_k$ stages there exists a value of $j_k = t_k - \lfloor R_k / \alpha_k \rfloor$, such that with reservation of $D_{j_k}^s$, the cost is minimum.

Proof: By lemma 1, $R_k < t_k.\alpha_k$. Earlier we assumed $f(j, k) = R_k - t_k.\alpha_k + j.\alpha_k$. Hence, $f(j, k)$ has a negative value for $j < t_k - R_k / \alpha_k$ and a positive value for $j > t_k - R_k / \alpha_k$.

As $\Delta E_{D_j^s, D_{j+1}^s} = (D_{j+1}^s - D_j^s)f(j, k)$ and $D_{j+1}^s - D_j^s \geq 0$, we can write $E_{D_1^s} \geq E_{D_2^s} \geq \dots \geq E_{D_{j_k}^s}$ and $E_{D_{j_k}^s} \leq E_{D_{j_k+1}^s} \leq \dots \leq E_{D_{t_k}^s}$ where $j_k = t_k - \lfloor R_k / \alpha_k \rfloor$.

The above result along with lemma 2 proves that cost is minimum when reservation is done for an amount of resources equal to $D_{j_k}^s$. \square

It is to be noted that the index of the demand in the sorted demand vector, for which the total cost is minimized, is independent of the values of demand vector.

So, from theorem 1 we can conclude that for a single type contract k , the optimal reservation is the j^{th} smallest element

in the demand vector where $j = t_k - \lfloor R_k / \alpha_k \rfloor$. That means we do not need to sort the demand vector since we can find the j^{th} smallest element in $O(t_k)$ time using randomized select [10] algorithm.

In situations where demand vector is not available for entire duration t_k , but for a smaller duration t'_k , we just optimize the cost over the duration t'_k . Let $j = t'_k - \lfloor R_k / \alpha_k \rfloor$. By using logic similar to the above, it can be shown that optimal reservation in this case is D_j^s , if $j > 0$ and it is zero, otherwise (j maybe negative for small values of t'_k). By lemma 1, $R_k < t_k \cdot \alpha_k$ but R_k may be greater than $t'_k \cdot \alpha_k$.)

6 HEURISTIC-BASED RESOURCE RESERVATION LEADING TO SUB-OPTIMAL SOLUTIONS

The heuristic-based resource reservation as discussed in Section 5 provides the optimal solution like IPP-based resource reservation under the assumption of having a single type contract k and $|D| \leq t_k$. In this section we extend the heuristic to more general case where these assumptions are relaxed to allow the following situations

- 1) The demand vector size can be more than the contract duration (i.e. $|D| > t_k$) and
- 2) More than one contracts/providers are available (i.e. considering multiple contracts).

We propose the following algorithms to provide sub-optimal solutions for such situations. First we consider single contract k with any demand duration and then we consider multiple contracts with any demand duration.

6.1 Single Contract Reservation

In this reservation strategy, only one contract (k) is considered at a time. The k^{th} contract is denoted by C_k which is a tuple (R_k, t_k, α_k) . The demand duration may be more than t_k . The reservation decision is taken for each contract duration (called segment) separately during the whole duration of the demand vector as shown in Fig. 3.

Let the demand vector D be available for stages $t = 1, 2, \dots, T$. We divide the whole duration of T into multiple segments. If T is divisible by t_k , then each segment is of duration t_k . Otherwise, for $i = 1, 2, \dots, \lfloor T/t_k \rfloor$ segment duration is t_k and the last segment is of duration $T - t_k \cdot \lfloor T/t_k \rfloor$. If we do not allow reservation made at two different stages to overlap, the optimal reservation for each segment is given by theorem 1 as shown in Section 5 and the amount to be reserved at intervals of t_k is given by Algorithm 1.

For each of the $\lceil \frac{T}{t_k} \rceil$ segments, we find the j^{th} smallest element of the segment of demand vector that is covered by that contract where $j = t_d - \lfloor \frac{R_k}{\alpha_k} \rfloor$ (t_d is t_k except possibly for the last segment). The decided amount of resources are reserved at the beginning of that segment.

6.2 Multiple Contracts Reservation

While Single Contract Reservation algorithm considers single type of contract, the reservation strategy presented here considers multiple types of contracts provided by the same CSP as well as different CSPs. Without loss of generality,

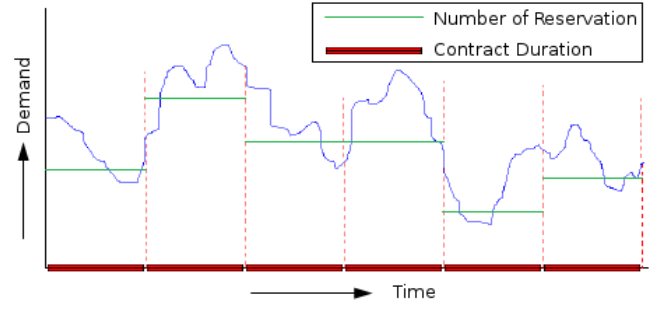


Fig. 3. Single Contract Reservation

Algorithm 1: Single Contract Reservation

input : Demand Vector $D[1..T]$, Contract C_k .
output: Reservation Decision in vector $x[1..\lceil \frac{T}{t_k} \rceil]$

begin

$T_r \leftarrow T$

$S \leftarrow 1$

$i \leftarrow 1$

while $T_r > 0$ **do**

$t_d \leftarrow \min(T_r, t_k)$

/ t_d is t_k except possibly for the last segment */*

$j \leftarrow t_d - \lfloor \frac{R_k}{\alpha_k} \rfloor$

$F \leftarrow S + t_d - 1$

if $(j > 0)$ **then**

$\lfloor x[i] \leftarrow j^{th} \text{ smallest element in } D[S \dots F]$

else

$\lfloor x[i] \leftarrow 0$

reserve $x[i]$ resources at stage S .

$T_r \leftarrow T_r - t_d$

$S \leftarrow S + t_d$

$i \leftarrow i + 1$

we can assume $t_1 > t_2 > \dots > t_K$ where t_k is the contract duration of k^{th} type of reservation contract. We make a realistic assumption that $R_1/t_1 < R_2/t_2 < \dots < R_K/t_K$ where R_k is the reservation charge of k^{th} type of reservation contract. If for example reservation cost of a one-year contract is not greater than one third of that of a three-year contract no one will ask for a three year contract.

As the reservation cost per hour is minimum for contract type 1, we would reserve as much as possible for the longest duration t_1 using Single Contract Reservation algorithm for contract type 1. We will repeat for the successive smaller duration contracts after updating the remaining demands. After considering all the available contracts, the remaining demands will be fulfilled from the on-demand resources.

7 IMPLEMENTATION AND EVALUATION

The algorithms for heuristic-based resource reservation have been implemented and their performances have been evaluated and compared with the optimal (IPP-based) one. On-demand

Algorithm 2: Multiple Contracts Reservation

input : Demand Vector $D[1..T]$, Contract Vector $C[1..K]$.

output: Reservation Decision in vector $x[1..K][1..max(\lceil \frac{T}{t_k} \rceil)]$

$/*C[i] = (R_i, t_i, \alpha_i)$ and $t_1 > t_2 > \dots t_K */$

begin

for $i \leftarrow 1$ **to** K **do**

$temp \leftarrow$

$Single - Contract - Reservation(D, C[i])$

$/* temp$ holds the reservation decision vector $*/$

for $j \leftarrow 1$ **to** $|temp|$ **do**

$/* Update the Remaining Demands */$

for $l \leftarrow 1$ **to** t_i **do**

$D[(j-1) \cdot t_i + l] \leftarrow$

$\max(D[(j-1) \cdot t_i + l] - temp[j], 0)$

and reserved pricing models offered by Amazon EC2 have been considered for this purpose. For reserved pricing model, both, 1-year and 3-years contracts have been considered. The demand vector of VMs have been parsed with the real traces of loads from four different *standard workload archives* of SDSC Blue Horizon, SDSC SP2, Sandia National Labs and High Performance Computing Center North(Sweden).

7.1 Experimental Setup

The IP formulation for this problem has been solved using *GLPK (GNU Linear Programming Kit) [9] Version 4.47*. The heuristic-based resource reservation algorithms have been implemented using *Java SDK v6*. Both the IPP and the proposed algorithms have been executed on a quad core Intel i5 2.4GHz processor with 4GB RAM.

As discussed earlier, IPP is NP-hard and in our experimental set-up it had not been possible to solve problems involving demand vector of size more than 6 months. Therefore, the Amazon EC2 contracts for 1-year and 3-years have been scaled down to 1-month and 3-months respectively. The associated costs have been scaled down using the following equations.

$$R_{1-month} = R_{1-year} / 12 \quad (7)$$

$$R_{3-months} = (R_{3-years} / 36) * 3 \quad (8)$$

It is easy to observe that the above two equations keep the hourly discount of reserved VM over on-demand VM unchanged.

We have used a single type of VM in our simulation. However, it can handle multiple types of VM also. For multiple types of VM, we need to determine the demands for each category of VMs separately and then we need to apply our heuristics to find the amount of reservation for each category. The properties of the VM we use in our simulation are listed in Table 2.

TABLE 2
Properties of the VM used in the Simulation

Property	Value
Type of VM	Standard large (linux)
Reservation Cost(1-month)	\$32.00
Reservation Cost(3-months)	\$20.25
On-demand Usage Cost	\$0.24/hour
Reserved VM (1-month) Usage Cost	\$0.136/hour
Reserved VM (3-month) Usage Cost	\$0.108/hour

7.2 Experimental Results and Discussion

Though the experiment has been carried out for four different sets of demand data, in this section we use the results of SDSC Blue Horizon and Sandia National Labs workload archives for the evaluation. The demand data derived from SDSC Blue Horizon workload archive has the property of uniform distribution of the demands as shown in Fig. 4. Whereas a non-uniform distribution of the demand is observed in the demand data derived from Sandia National Labs workload archive as shown in Fig. 5.

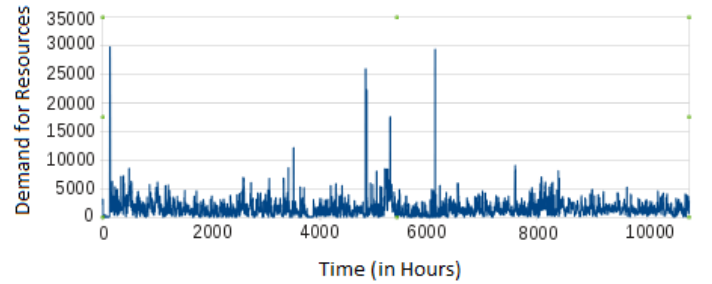


Fig. 4. Uniform Demand Data of SDSC Blue Horizon

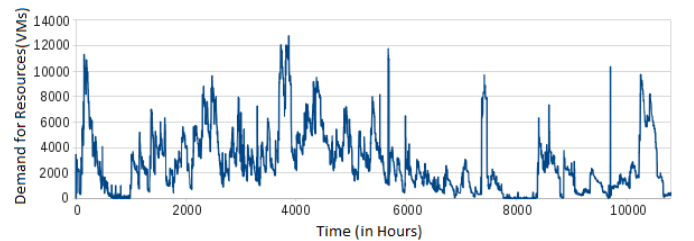


Fig. 5. Non-Uniform Demand Data of Sandia National Lab

Minimizing the total cost of deploying and running an application in cloud is the primary objective of optimal resource reservation problem. So we use *Total Cost* and *Cost Increase* as the metrics for comparing the results of different reservation strategies. The metric *Total Cost* includes the one time reservation cost and the usage costs of on-demand VMs as well as reserved VMs to meet the demands for a particular duration. Whereas the metric *Cost Increase* measures the percentage increase in the total cost in comparison with the Optimal (IPP) strategy to meet the demands of VMs for a particular duration. Apart from Total Cost and Cost

Increase, we also use the metric *Overhead* to show the total computational time needed by different reservation strategies under the experimental environment. This metric is useful to analyse the feasibility of a particular reservation strategy.

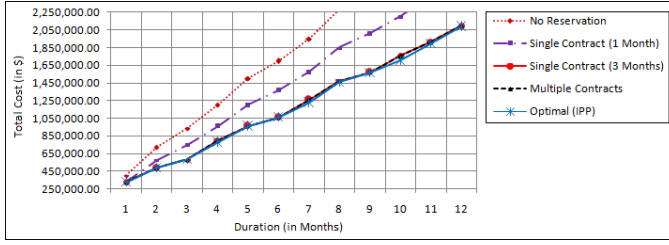


Fig. 6. Total Cost for SDSC Blue Horizon

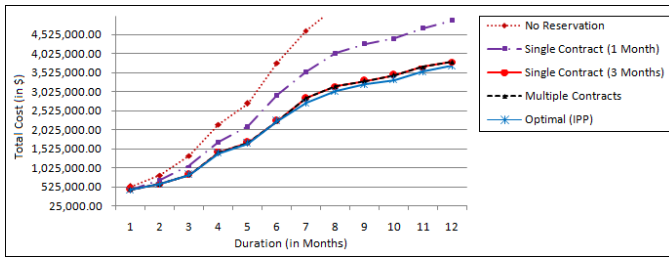


Fig. 7. Total Cost for Sandia National Lab

Total costs for different duration, ranging from 1 month to 12 months, are shown in Fig. 6 for the SDSC Blue Horizon workload archive and in Fig. 7 for the Sandia National Labs workload archive. We have compared the total costs for optimal strategy (IPP), heuristic-based strategies with multiple contracts, with single contract for 1 month and with single contract for 3 months. Total cost without any reservation is also depicted in the figures.

From the above figures, following observations are made:

- 1) While comparing with *no reservation*, it has been observed that all the reservation strategies reduce the total cost significantly.
- 2) Contracts with longer duration provide reduced costs compared to the contracts with shorter duration. For example, heuristic-based reservation algorithm for single contract of 3-months outperforms single contract of 1-month reservation. This is because, for any two contracts i and j such that $t_i < t_j$, $\frac{R_j}{R_i}$ is always less than $\frac{t_j}{t_i}$.
- 3) Total costs of heuristic-based reservation for multiple contracts and single contract for 3 months are very close to the total costs of optimal (IPP) reservation strategy.
- 4) The relative performance of different reservation strategies are unaffected by the distribution (uniform or non-uniform) of the demand data.

The percentage increases in the total costs in case of different heuristic-based strategies over the optimal (IPP) strategy are tabulated in Table 3 for SDSC workload archive. Clearly, Single Contract (1 month) strategy is not acceptable. However, Single Contract (3 months) as well as Multiple Contracts perform well and the average increase in the total costs is below 1%. The relative increases in costs for both, uniform

TABLE 3
Cost Increase (in %) of Different Resrvation Strategies compared to Optimal(IPP)Strategy

Duration (months)	1	2	3	4	5	6
No Reservation	23.61	47.88	59.95	53.50	56.00	60.05
Single Contract (1-month)	1.59	19.25	28.30	23.32	25.25	29.23
Single Contract (3-months)	1.14	0.00	0.00	1.90	0.10	0.00
Multiple Contracts	0.93	0.01	0.00	1.82	0.10	0.00

Duration (months)	7	8	9	10	11	12
No Reservation	58.71	56.00	57.75	59.18	58.15	59.62
Single Contract (1-month)	28.20	26.82	28.58	29.40	28.27	29.11
Single Contract (3-months)	2.78	0.92	0.26	3.18	1.22	0.36
Multiple Contracts	2.74	0.92	0.26	3.15	1.22	0.36

and non-uniform demand data, are shown in Fig. 8. Following observations are made from the graph:

- 1) Increase in cost is higher for non-uniform demand data compared to uniform demand data.
- 2) For longer duration, the cost increase is higher due to cumulative effect.
- 3) The cost increase tends to 0% for durations which are multiple of 3.
- 4) The cost increase is maximum for durations which are multiple of 3 + 1 (e.g. 1, 4, 7 etc) and gradually decreases till the next duration which is multiple of 3.

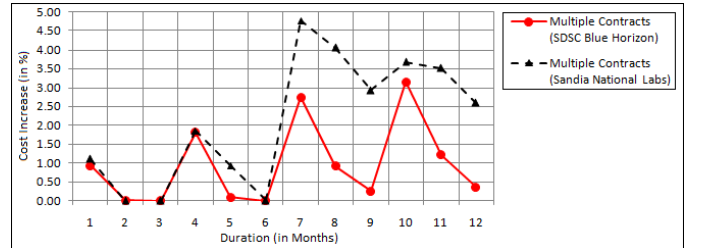


Fig. 8. Percentage of Increase in Total Cost Relative to Optimal (IPP)

From the above analysis, it may be concluded that the heuristic-based reservation for multiple contracts or single contract with longer duration (in this case 3 months) may be used up to certain demand duration (in this case 6 months - which actually will be 6 years if scaling down effect is removed). If demand duration is longer (here longer than 6 months), the cost increase in comparison with the optimal strategy is much higher, and hence the effectiveness of the heuristic-based solution is reduced.

Finally, Table 4 shows the effectiveness of our proposed heuristic-based reservation strategies in terms of the metric *Overhead*. While Single Contract (3 months) and Multiple Contracts require 1113 μ s and 2220 μ s respectively for 12 months demand duration, the Optimal strategy requires

$7 \times 10^5 \mu s$ for only 1 month demand duration. As expected and analyzed in Section 5, Table 4 shows that the heuristic-based reservation strategies have a linear time complexity compared to the exponential time complexity of the optimal strategy.

TABLE 4
Overhead (in μs) of Different Reservation Strategies

Duration (months)	1	2	3	4	5	6
Single Contract (3-months)	271	463	535	582	659	710
Multiple Contracts	613	913	1011	1170	1303	1468
Optimal (IPP)	7 x 10^5	25 x 10^5	51 x 10^5	101 x 10^5	218 x 10^5	367 x 10^5
Duration (months)	7	8	9	10	11	12
Single Contract (3-months)	793	863	924	1024	1083	1113
Multiple Contracts	1615	1726	1861	1988	2130	2220
Optimal (IPP)	635 x 10^5	776 x 10^5	1005 x 10^5	1228 x 10^5	1654 x 10^5	1965 x 10^5

8 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a heuristic-based algorithm to solve the cost optimization problem of cloud resource provisioning in linear time. We have mathematically proved that the heuristic provides optimal solution under some restrictions. We then extend the heuristic to cover the situations where these restrictions are removed. The experimental results show that the solution is very close to the optimal one with minimal overhead. Even though the IPP based approach gives the best solution for the problem, the total number of variables becomes too large if demand data is of duration more than six months and thus become impossible to solve by existing softwares like *GLPK*. At the same time, the linear time heuristic makes it possible to work on hourly demand data of length 3 years or more without any difficulty. We can conclude that the proposed heuristic based approach help to overcome the drawback of IPP based method sacrificing very little on the total cost incurred.

To simplify the heuristic, we have ignored the uncertainty of demands and prices in this work. Future work in this area could be how to bring those uncertainties into consideration. We would also like to include the option of spot pricing scheme to minimize the total cost further.

REFERENCES

[1] Amazon EC2 Pricing Schemes. Available from: <http://aws.amazon.com/ec2/purchasing-options/>

- [2] Amazon EC2 REserved Instances. Available from: <http://aws.amazon.com/ec2/reserved-instances/>
- [3] S. Khatua, A. Ghosh and N. Mukherjee, *Application-centric Cloud management*. In 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA) 2011, pp. 9-15, doi: 10.1109/AICCSA.2011.6126627
- [4] R. Buyya, et al, *Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities*. In 10th IEEE International Conference on High Performance Computing and Communications (HPCC 08) 2008, pp. 5-13, doi: 10.1109/HPCC.2008.172
- [5] I. Foster, Y. Zhao, I. Raicu and S. Lu, *Cloud Computing and Grid Computing 360-Degree Compared*. In Grid Computing Environments Workshop (GCE '08), 2008, pp. 1-10, doi: 10.1109/GCE.2008.4738445
- [6] G. Juve and E. Deelman, *Resource Provisioning Options for Large-Scale Scientific Workflows*, In 4th IEEE International Conference on eScience (eScience '08), 2008, pp. 608-613, doi: 10.1109/eScience.2008.160
- [7] K. Miyashita, K. Masuda, and F. Higashitani, *Coordinating Service Allocation through Flexible Reservation*. In IEEE Transactions on Services Computing, vol. 1, no. 2, pp. 117-128, 2008.
- [8] J. Chen, G. Soundararajan and C. Amza, *Autonomic Provisioning of Backend Databases in Dynamic Content Web Servers*. In IEEE International Conference on Autonomic Computing (ICAC '06), 2006, pp. 231-242, doi: 10.1109/ICAC.2006.1662403
- [9] GNU Linear Programming Kit (GLPK). Available from: <http://www.gnu.org/software/glpk/>
- [10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*. The MIT Press, Third Edition, 2009, ISBN: 978-0262033848
- [11] W. Deyuan, W. Ying, L. Jichun, X. Ke, L. Wenjing Li and Q. Xuesong, *Pricing reserved and On-Demand Schemes of cloud computing based on option pricing model*, In 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2013, pp. 1-3.
- [12] M. Mazzucco and M. Dumas, *Reserved or On-Demand Instances? A Revenue Maximization Model for Cloud Providers*. In IEEE International Conference on Cloud Computing (CLOUD), 2011, pp. 428-435, doi: 10.1109/CLOUD.2011.25
- [13] L. Qian and G. Yike, *Optimization of Resource Scheduling in Cloud Computing*, In 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010, pp. 315-320, 2010 doi: 10.1109/SYNASC.2010.8
- [14] H. Menglan, L. Jun and B. Veeravalli, *Optimal provisioning for scheduling divisible loads with reserved cloud resources*, In 18th IEEE International Conference on Networks (ICON), 2012, pp. 204-209, doi: 10.1109/ICON.2012.6506559
- [15] S. Chaisiri, L. Bu-Sung and D. Niyato, *Optimization of Resource Provisioning Cost in Cloud Computing*, In IEEE Transactions on Services Computing, vol.5, no.2, pp. 164-177, 2012 doi: 10.1109/TSC.2011.7
- [16] T. S. MahmoudLoad, D. Habibi, O. Bass and S. Lachowicz, *Demand Forecasting: Model Inputs Selection*